DOCUMENT RESUME

ED 198 809 IR 009 178

AUTHOR Rumelhart, David E.; Norman, Donald A.
TITLE Analogical Processes in Learning. Technical Report, April 1979-August 1980.
INSTITUTION California Univ., San Diego. Center for Human Information Processing.
SPONS AGENCY Office of Naval Research, Washington, D.C. Personnel and Training Branch.
REPORT NO CHIP-TR-8005
PUB DATE Sep 80
CONTRACT N00014-79-C-0323
NOTE 38p.

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS *Cognitive Processes; Diagrams; *Epistemology; *Learning Processes; *Learning Theories; Memory; Methods; Models
IDENTIFIERS Analogies; Schemata

ABSTRACT

Human knowledge consists of schemata based on the specialized procedures used to interpret events in the environment. New schemata are created by modifying models of existing schemata, i.e., learning by analogy. This means that a new situation will be interpreted in accordance with the schema most similar to it in the learner's repertoire. If the schema is an inappropriate fit, a new one based on the original model will be created. By carrying over existing features of a current schema, we can make inferences about a new situation without explicit knowledge of that situation. If the analogy is a good one, most of the inferences from the original schema or source domain will be appropriate to the new one. Conceptual models should have the following properties: (1) it should be based on a domain about which the student is knowledgeable, (2) the target domain and source domain should differ by a minimum number of dimensions, (3) operations natural within the target domain should also be natural in the source domain, and (4) operations inappropriate within the target domain should also be inappropriate within the source domain. Examples from several domains include turtle geometry, kinship terms, and learning a computer text editor. (Author/EK)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U S DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

ED198809

ANALOGICAL PROCESSES IN LEARNING

David E. Rumelhart & Donald A. Norman

Program in Cognitive Science
Center for Human Information Processing
University of California, San Diego
La Jolla, California 92093

Copyright © 1980 David E. Rumelhart & Donald A. Norman

To be published in Anderson, J.R. (Ed.), Learning and Cognition. In preparation.

This research was sponsored by the Personnel and Training Research Programs, Psychological Sciences Division, Office of Naval Research, under Contract No. N00014-79-C-0323, Contract Authority Identification Number, NR 157-437. Approved for public release; distribution unlimited.

IR009178

Introduction

One of the more lamentable results of the information processing revolution within psychology over the past twenty years has been the replacement of the term learning by the term memory. Whereas it is sometimes difficult to distinguish the learning experiments of twenty years ago from today's memory experiments, it is increasingly clear that remembering is only one kind of learning. As long as our theories of knowledge representation were simple, this substitution caused no problem. If knowledge is essentially declarative and unstructured, new learning can be carried out by simply adding new facts to the data base. Over the past several years, however, we have been led to a significantly more complex representational theory. In particular, we have come to see knowledge as embedded in schemata which we see as largely composed of specialized bits of procedural knowledge (c.f. Bobrow & Norman, 1975; Rumelhart & Ortony, 1977; Rumelhart, in press). In a recent paper (Rumelhart and Norman, 1978), we began a logical analysis of what learning must amount to in the context of a schema based representational system. According to our analysis, the adoption of the schema as the basic unit of knowledge representation has implicit in it three qualitatively different kinds of learning. These are:

(1) Accretion-- the encoding of new information in terms of existing schemata. On our view, new information is interpreted in terms of relevant preexisting schemata and some trace of this interpretation process remains after the processing is complete. This trace can serve as the basis for a later reconstruction of the original input. Thus, processing information changes the system, giving it the ability to answer questions it could not have previously answered. The system has thereby learned something new. This is presumably the most common and least profound sort of learning. Note, that no new schemata are involved in this sort of learning. An organism which learned only in this way could never gain any new schemata; all learning would be in terms of instantiations of already existing schemata.

(2) Tuning or Schema Evolution-- the slow modification and refinement of a schema as a function of the application of the schema. Schema evolution is presumably a central mechanism in the development of expertise. With experience, an existing schema can be slowly modified to conform better and better to the sorts of situations to which it is to apply.

(3) Restructuring or Schema Creation-- the process whereby new schemata are created. This kind of learning, which we have

called restructuring, or more recently simply structuring, involves the creation of new schemata which, through tuning, can themselves become highly refined and distinct concepts.

Our models of memory are thus models of learning by accretion. Many such models exist. It is substantially more difficult to create models of learning of the other two types. Therefore, we have begun to focus our attention on the processes of schema creation and schema evolution. In this paper, we report some of the theoretical and empirical approaches we have taken to the study of schema creation. We begin with a discussion of knowledge representation and show why we believe learning to be central and why we believe analogy is such an important mechanism of learning. Then, we will describe a simple model of how new schemata might be formed by analogy. Finally, we describe an empirical situation in which we think we find evidence for such learning and show how our model might generate the results we have observed.

Some Characteristics of the Human Knowledge Representation System

Since the issue of knowledge representation has played a central role in our thinking about learning, it is useful to begin our discussion with a few observations on some of important characteristics of knowledge representation. It is, of course, cliche that it is impossible to evaluate a representational system apart from the process which operates on it. Consequently, in modeling any cognitive process, there is always the problem of deciding how to partition that part of the knowledge system which is "process" from that part which is "data." Depending on the relative amounts of the system allocated to "process" and to "data", we have what Winograd (1975) has called "procedural" or "declarative" representational systems. Some authors have emphasized the "data", trying to have as few special purpose procedures as possible; such a system is called declarative. Others have emphasized the processes involved and have largely embedded the knowledge of the system within these processes. These systems are generally called procedural. The issues involved in choosing one or the other of these strategies has been described by Winograd as the "declarative-procedural controversy." In his paper on this topic, Winograd (1975) offered a useful analysis of the topic. We summarize the issues briefly below.

On the one hand, there are facts. It is often quite convenient to conceptualize the contents of memory as a set of facts and to imagine retrieval from memory to be the application of general, content free retrieval processes. With this view, reasoning can be conceptualized as the production of inferences based on these facts. Of course, a representational system such as this requires rules of inference separate from these "facts", but these rules are conceptualized as very general and in no way tied to the specific content of the facts to which they apply. Here, the best analogy is between the axioms and theorems of a mathematical system on the one hand (the facts) and the rules of inference of that system on the other (the processes). Once the rules of inference are specified, the axioms can be changed at will and the

system will still continue to produce correct inferences.

On the other hand, there are operations. It is often convenient to construct special purpose procedures which have special knowledge of the various contingencies of use built into them. All systems must have some operations. Procedurally based systems consist primarily of such special operators.

In his comparison of these two representation types, Winograd notes four basic characteristics on which the two kinds of representational systems typically differ.

(1) Flexibility. Within a declarative system, the same fact can be used whenever it is relevant. Once a fact is added to the data base, it is available for use by any of the inference rules. In a procedural system, with knowledge contextually embedded, relevant information may be known but not available. Because it is stored implicitly, as part of a procedure, independent access to the knowledge is impossible. In a declarative system, on the other hand, knowledge does not have to be specified differently for each context in which it may be needed.

(2) Learnability. It is easy to add new information to a declarative system. A new statement (or axiom) or even an entirely new domain of knowledge can be added to the data base and new inferences automatically become possible without the addition of any new rules of inference. In procedural representations, the procedures are generally hand crafted by the theorist and it is difficult to see how new procedures could be evolved. Moreover, since what is general and what is specific about procedural representations are not often easily separated, there is little or no transfer from one domain to another. In short, the process whereby new knowledge is added to procedurally based systems is enormously more difficult than adding new knowledge to declarative systems.

(3) Accessibility. Knowledge separated out in the form of a set of discrete statements is relatively easy to find and express as isolated entities. Knowledge stored in a more procedural, context dependent fashion is impossible to separate from the contexts in which it is employed. Knowledge which is relatively easy to express is taken to be stored declaratively whereas knowledge which is known only tacitly is taken to be procedural.

(4) Efficiency. Procedural representation systems have the advantage of efficiency. With general inference rules, care must be taken to "handle" even the most obscure cases. With procedural representations, however, specific aspects of the problem domain can be taken directly into account in the procedures. It is therefore possible to employ heuristics which

> might fail in general, but work in specific cases. This allows for the very direct solution of problems for which the system is best tuned but perhaps no solutions at all for problems outside that domain. In practice, the ability to "get away with" limited but efficient solutions makes it much easier to specify a knowledge system that works at all.

In many ways it seems that humans have more of the characteristics attributed to procedural systems than those attributed to declarative ones. Our ability to reason and otherwise use our knowledge appears to depend strongly on the context in which that knowledge is required. Most of the reasoning we do apparently does not involve the application of general purpose reasoning skills. Rather, it seems that most of our reasoning ability is tied to particular bodies of knowledge.

Perhaps the classical case of using knowledge how (procedural knowledge) to produce knowledge that (factual knowledge) occurs in the domain of grammatical judgements. The knowledge that we have about language seems to be largely embedded in the procedures involved in the production and comprehension of linguistic utterances. This is evidenced by the relative ease with which we perform these tasks when compared with our ability to explicate the knowledge involved in them. Semantic knowledge would appear to be the same. Whereas we can quickly interpret sentences, it is only with the most painstaking effort that we can produce definitions of terms with any generality.

Perceptual knowledge is even more plausibly viewed as knowledge how. Whereas we all know a dog when we see one, it is very difficult to sort out exactly what we look for in making our judgement. We know how to tell a dog without knowing how we know it. Similarly, we know how to perform many skills (e.g. playing tennis), but it is rather difficult to access the facts on which this knowledge is based. Thus, it seems useful to imagine knowledge such as this to be in the form of procedures or programs for doing these activities. The knowledge that we have is implicit--somehow tied up in the operations in which we actually use that knowledge.

One nice demonstration of this comes from the work of Wason and Johnson-Laird (1972) and some more recent replications and extensions of their work carried out by Roy D'Andrade. [1] Subjects in D'Andrade's experiments were given one of two formally equivalent problems to solve. Half of the subjects were given the task illustrated in the left portion of Figure 1. Subjects were shown the four cards illustrated in the Figure and told that:

> All labels made at Pica's Custom Label Factory have a letter printed on one side, and a number printed on the other side.

---

1. Roy D'Andrade has kindly given us access to the data from his as yet unpublished experiment.

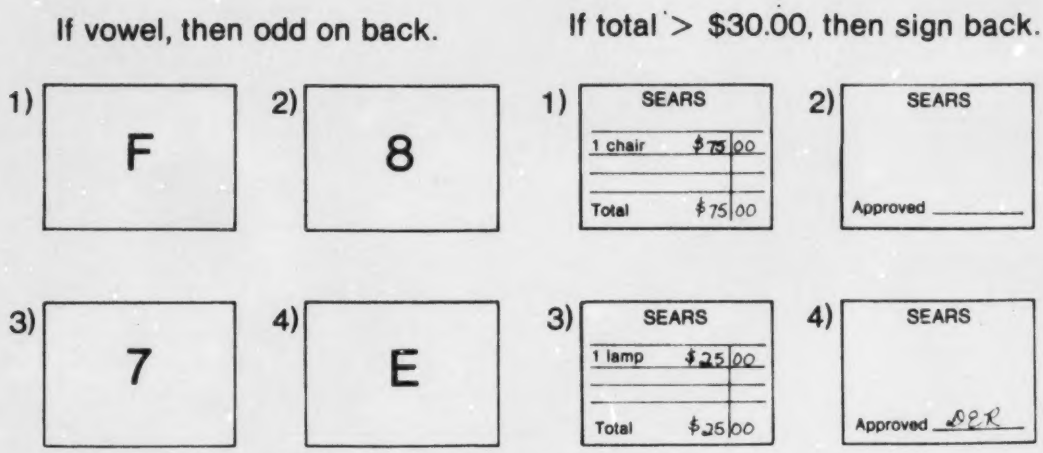

Figure 1. Stimuli for the two conditions of D'Andrade's reasoning experiment. The left panel shows the stimuli for Label Factory condition. The right panel shows the stimuli for Sears store condition.

> As part of your job as a label checker at Pica's Custom Label Factory, you have the job of making sure that all labels with a vowel printed on one side have an odd number printed on the other side. Which of the labels ... would you have to turn over to make sure the label was printed correctly?

Only 13 percent of the subjects correctly indicated that cards 2 and 4 (the cards marked with an 8 and an E, respectively) must be checked. Card 4 must, of course, be checked because it may have an even number on the back. Card 2 must also be checked since it might have a vowel on the back and thus violate the rule. No other cards must be checked.

It has been argued that this is a case of our interpreting the simple conditional as a bi-conditional. In any case, results like these are often taken to illustrate the weakness of the human reasoning system. However, the results of the second part of D'Andrade's experiment point up the fallacy in this conclusion. The right panel of Figure 1 illustrates the stimuli for these subjects. Subjects were told that:

> As part of your job as an assistant at Sears, you have the job of checking sales receipts to make sure that any sale of over $30.00 has been approved by the section manager. (This is a rule of the store.) The amount of the sale is written on the front of the form. Which of the forms ... would you have to turn over to make sure the sales clerks had followed the regulation?

In this case nearly 70 percent indicated the correct forms forms 1 and 2, the $75 and the unsigned forms. Formally, these two problems are identical. Yet, when phrased in terms of the familar setting of the Sears store, over five times as many subjects were able to correctly solve the problem.

What is the difference here? Why do people appear not to understand the meaning of "if" in the first case and understand it nearly perfectly in the second? This is exactly the kind of effect expected if our knowledge is embedded in a relatively inaccessible procedural format rather than as general rules of inference. The first case of the label factory represents a relatively unfamiliar case in which we can not rely on specific knowledge and must, therefore, rely on general reasoning processes. The second case more nearly approximates our "real life" problem solving situations. Once we can "understand" the situation, the conceptual constraints of our specific knowledge can be brought into play, and the problem readily solved. It is as if our knowledge representation already contains all of the reasoning mechanisms ordinarily required.

Thus, it would appear that the context dependencies inherent in the more procedural representational systems are also present in the human reasoning system. Similar results can be observed by watching a person attempt to learn a new body of knowledge. When we attempt to teach a child a new domain, we do not, in general, present it as an abstract

piece of new knowledge. Rather, we carefully instruct the child using the knowledge already tacitly available to "get across" the concept in question.

Consider, for example, how we teach children the concept of a fraction. Most curricula use the "pie" analogy. One half corresponds to one piece of a pie which has been cut down the middle. One fourth corresponds to one piece of a pie cut into four equal pieces, etc. Here, the teacher is taking advantage of the child's spatial intuitions to teach the abstract notions of a fraction. This analogy is very useful; upon learning it, the reasoning and problem solving strategies implicit in his knowledge of "pies," operations that can be performed on them etc., can be carried over into this abstract domain. The child can see that two quarters make a half, that if you have a whole and take away one quarter, you have three quarters remaining etc. The child needn't know how he knows this. These inferences are simply implicit in the analogy.

However, as with all analogies, the analogy is not perfect. Sometimes operations are required in the target domain (in this case with fractions) which are difficult or unnatural within the domain of the analogical source. Thus, whereas addition and subtraction of fractions is natural within the "pie" analogy, multiplication and division of fractions is unnatural and difficult to conceptualize. How do you take one piece of pie times another, or worse yet, how do you divide one piece of pie into another.

Fractions are sometimes taught through a different analogy. Once a child has learned multiplication and division, fractions can be understood as operations. A fraction is a compound operation. A fraction is merely a multiplication and a divide. Thus, one half of a number is that number multiplied by one and divided by two. Similarly, three fourths of a number is that number multiplied by three and divided by four, etc. Those taught by the operation method find the multiplication and division of fractions a very natural extension of their conceptualizations. One can, of course, readily do a "multiply and divide" of a fraction and produce a new fraction. These children, however, often find addition and subtraction of fractions very difficult. How do you add one "multiply and divide" to another?

Thus, depending on which of the two systems of analogies are tapped by the curriculum in question, the sorts of difficulties a child will have is predictable. If a child is taught through the "pie" analogy, he or she finds the addition and subtraction of fractions relatively natural. These are operations carried rather directly from the original "pie" domain. Multiplication and division of fractions, on the other hand, are often very difficult for these children.

Here again, it appears, that knowledge of fractions is best not thought of as a list of facts, but rather as a set of procedures we have learned. Moreover, these procedures are apparently not created de novo, but are generated through a systematic mapping of prior, often only

implicitly known, knowledge. Curriculum developers are always on the lookout for the perfect analogy. The perfect analogy is one in which the learner is already able to reason within the source domain with ease and in which all of and only the operations of the target domain are represented in the source domain. Needless to say, such domains are rare. Two kinds of diagnostic problems often arise. First, learners will have great difficulty in learning operations not implicit in the original source domain. This is illustrated by the example above. Secondly, learners will often carry features of the source domain incorrectly into the target domain. We will discuss an example of this later. Both of these examples are useful to the analyst for it is through these kinds of errors that we can find evidence of the analogical nature of the learning.

As yet another example of using knowledge how to derive knowledge that, consider the task of remembering the number of windows in your house. Most people report systematically "going through" the rooms in their house and "counting the windows". Clearly, in these cases, the knowledge of our windows is implicit in another body of knowledge. We can, however, derive this implicit knowledge by using our ability to imagine the rooms of our house systematically. Note, we know how to imagine the rooms of our house and make use of that ability to know that we have so and so many windows in our house. 2

To push this view perhaps harder than it ought to be pushed, it may well be that we "know" the alphabet by virtue of our knowing how to recite it. Although this may seem silly at first glance, it is certainly plausible that we "know" the identity of the letter before the letter before 'k' by virtue of our ability to recite the alphabet.

The human system does differ from existing procedural systems in one important way, however. The human system is notoriously adaptive. We are capable of applying knowledge learned in one domain to another; we are capable of readily learning new concepts and modifying old ones. Mimicking this flexibility has been the major problem for the procedural representational systems. It has proved rather difficult to build moderately general self modifying procedures.

For the past several years, we have been involved in the development of a representational system which combines the important aspects of the procedural and declarative structures in somewhat different ways (c.f. Rumelhart & Norman, 1973; Norman, Rumelhart & LNR, 1975). In our representational system, dubbed the Active Semantic Network, we have combined the declarative advantages of semantic networks with the procedural convenience of LISP-like languages. We developed a representational system in which a LISP-like interpreter operates directly on semantic networks (rather than lists) to perform its operations. In

---

2. Note, this example has occasionally been used to demonstrate the visual characteristic of our knowledge. It would seem to better illustrate how much of what we "know" is embedded in what we can "do."

this system, procedures are encoded as configurations of links in a semantic network. Whenever we treat a piece of network as a procedure, we employ a general interpreter which produces various outputs and modifications of the network. During these times, the fact that the procedures are themselves encoded in the network is irrelevant. These procedures could equally well be entirely external to the data base. However, since the procedures are encoded in the data base, they can, on occasion, be interrogated by other procedures. This allows procedures to be modified, retrieved, compared, deleted and otherwise operated on as only declarative data normally can be.

Although this conception has been a part of our representational system for some time, in practice (like most LISP structures) pieces of semantic net have either always been treated as data or have always been treated as procedures. The one exception to that was the work of Scragg (1975) who proposed a system that "looked through" a set of procedure definitions in order to answer hypothetical questions about what might happen if certain of those procedures were carried out.

In our recent work, we have leaned more and more heavily on the procedural view of our data structures, and the fact that they can also be viewed as semantic networks has been less and less important. We have argued that schemata (c.f. Rumelhart & Ortony, 1977; Rumelhart & Norman, 1978) are procedures which scan the input for information relevant to whether aspects of the input could represent instances of the concept represented by the schema. In doing this, the internal structure of the schema is irrelevant. The important question has been the operation of the schema, not its internal structure.

The internal structure of the knowledge representation is important when old knowledge must be applied to domains beyond that which it was originally designed to represent, when new knowledge must be assimilated and when pieces of knowledge must be compared. In short, it is under these conditions that the purely procedural perspective is inadequate and the knowledge must be viewed declaratively. We believe that the most common way in which people apply knowledge learned in one domain to another one is through analogical reasoning. We believe that the border between the procedural perspective and the declarative perspective can be usefully spanned by developing a mechanism for specifying new procedures based on the structure of old ones.

New Schemata by Analogy with Old

We thus propose a representational system in which all of the data can be viewed as either data or process. Such a system captures many facts about human knowledge in a natural way. We propose that all knowledge is properly considered as knowledge how but that the system can sometimes interrogate this knowledge how to produce knowledge that The means whereby this knowledge is extended is, we believe, best viewed as an analogic process similar in form to that proposed by Moore and Newell (1973). Just as new concepts in MERLIN are defined as old ones with certain specified differences, one can define new schemata as

systematic modifications on old ones.

The basic scheme whereby this may be done can be illustrated in terms of some very simple examples. Imagine that our knowledge of how to draw a square were embedded in the following simple turtle geometry program for drawing a square:

```
define square(:x)
     loop(4, &(forward(:x),right(90))).
```

This procedure would be represented within our Active Semantic Networks as shown in Figure 2. In this representation, terminal nodes represent either constants or variables whereas non-terminal nodes represent subprocedure names. Each branch on a tree represents an argument of a procedure. The left-most branch represents the first argument, the right-most one the last argument. Intermediate branches represent intermediate arguments. It is useful to observe that along with the conceptually important concepts of there being four sides and that the angles are 90 degrees, there are a number of "technical" aspects of the procedure needed in order to make it actually work out and be properly interpreted by the interpreter. In particular, there is LOOP which counts out the number of sides, there is the "&" which combines FORWARD and RIGHT into a single argument for LOOP.

This program successfully draws squares, and for most purposes the fact that it has the particular internal representation that it does makes no difference. It represents a kind of "knowledge how." Now consider what a similar sort of program to draw a pentagon might be like.

```
define pentagon(:x)
     loop(5,&(forward(:x),right(72))).
```

Figure 3 shows the network representation of this procedure. A comparison of figures 2 and 3 shows the similarity of structures of these two procedures. Note that all of the basic bookkeeping and technical aspects of the two procedures are identical. They differ only in the fundamental ways pentagons and squares differ, that is, in terms of the number of sides (five instead of four) and of the angles through which the turtle must turn in order to draw the figure (72 instead of 90 degrees). It should be clear that this new procedure, the pentagon procedure, could readily be made by copying the structure of the square procedure and replacing the constant 4 by the constant 5 and the constant 90 by the constant 72. We see this as the fundamental process of learning by analogy, taking one schema and creating another one identical to it except in specified ways.

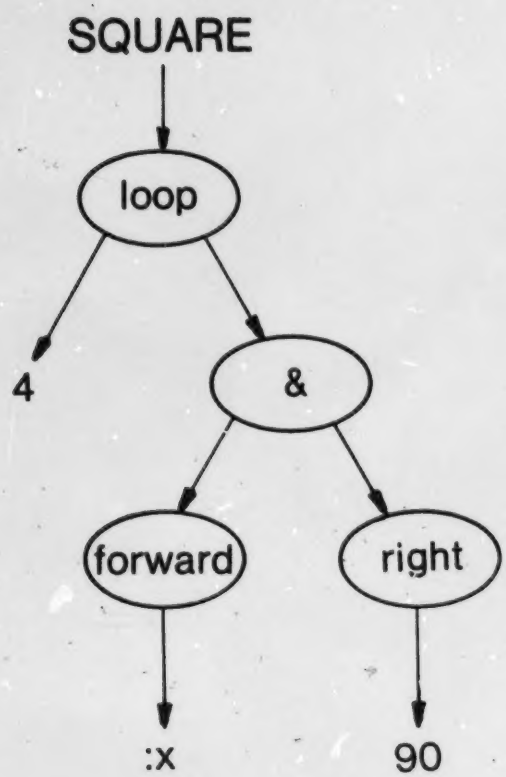

Figure 2. The Active Semantic Network representation of SQUARE, a procedure for drawing squares. Terminal nodes represent either constants or variables. Nonterminals written in ovals are subprocedure names. Arcs represent the arguments of the subprocedures.

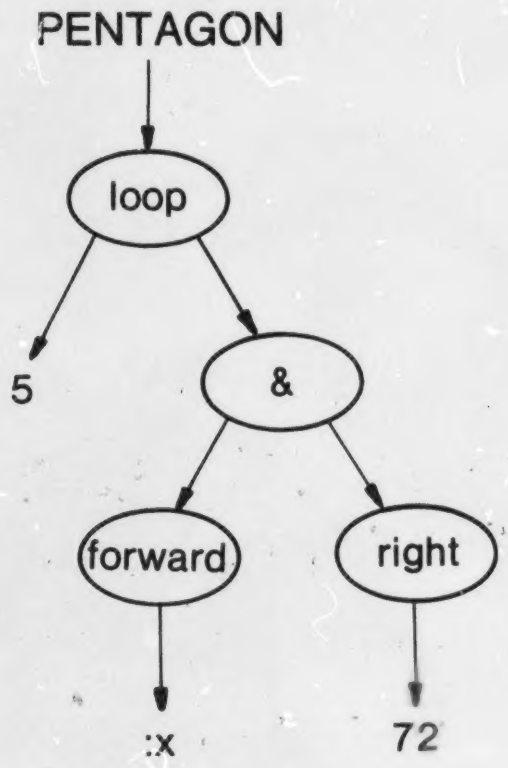

Figure 3. Active Semantic Network representation of PENTAGON, a procedure for drawing pentagons.

We have implemented this process within our computer simulation program with a program we call IS-LIKE. The statement:

pentagon is-like "square" with 5 for 4 and 72 for 90.

causes the program pentagon to be created. It is important that all of the hand-crafted aspects of SQUARE are automatically brought into the structure of PENTAGON without any need for special knowledge of what these structures are. Of course, the same procedure could readily be applied to generate an OCTAGON or any other regular polygon we might wish. In fact, the statement

regular-polygon is-like "square" with :n for 4 and ratio(360 to :n) for 90.

will generate the structure illustrated in Figure 4 which will draw any regular polygon. In general, the "is-like" program can generate any new procedure in which every occurrence of a particular constant or variable is replaced by another constant, another variable, or a subnetwork or in which every occurrence of a particular subprocedure is replaced by another. This last point is illustrated in the following discussion.

Note that the PENTAGON and the SQUARE procedures are completely distinct; changes made in SQUARE after PENTAGON has been generated will not be transfered to PENTAGON. However, the lineage of PENTAGON remains in the incidental aspects of the way it draws its pentagon. In particular, both SQUARE and PENTAGON construct their respective figures in a clockwise fashion, turning right at every corner. If it were important, we could readily create a LEFT-SQUARE which generates it's figure in the opposite direction by replacing the occurrences of the subprocedure RIGHT with the subprocedure LEFT. Thus, the statement:

left-square is-like "square" with "left" for "right".

will create a procedure which draws its figure in a counter clockwise direction. The network representation for LEFT-SQUARE is identical to SQUARE except that the non-terminal node for RIGHT is replaced by one for LEFT.

There are additional aspects of this scheme of creating new schemata through analogy to old ones which require a somewhat richer domain to illustrate. Thus, consider the domain of kinship relations. Imagine a system in which the basic kinship relations are stored in a network like the one illustrated in Figure 5. It is possible to represent all of the possible kinship relations of English in terms of the five basic relations illustrated in the figure--namely, "child", "parent", "spouse", "male", and "female". The figure is supposed to represent the fact that "Mary" is the daughter of "Alice", that "Maggie" is the grandmother of "Alice" and that "Alice" and "Henry" are married.

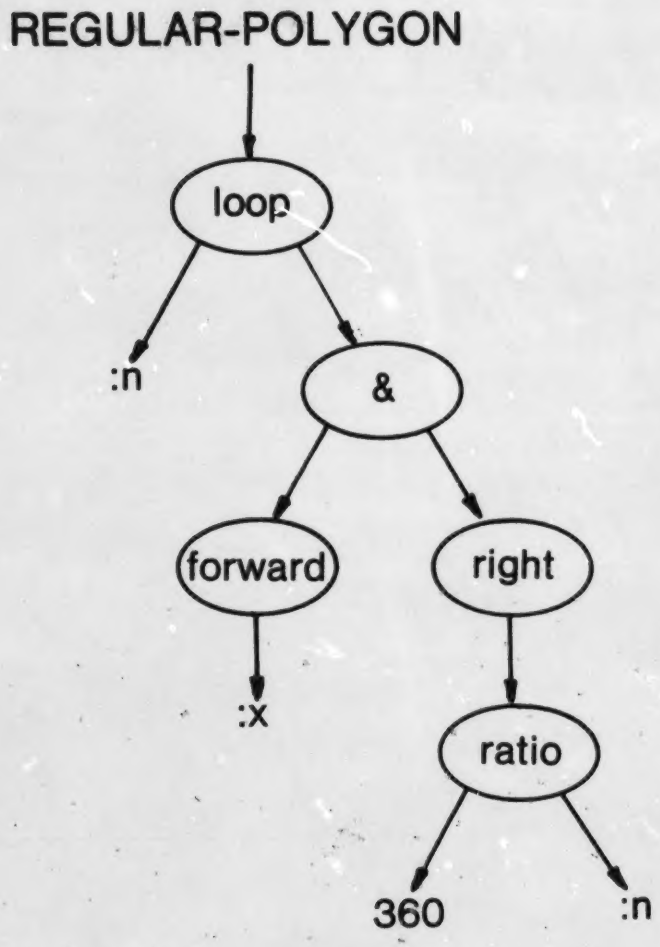

Figure 4. Network representation for REGULAR-POLYGON, a procedure for drawing a regular polygon.

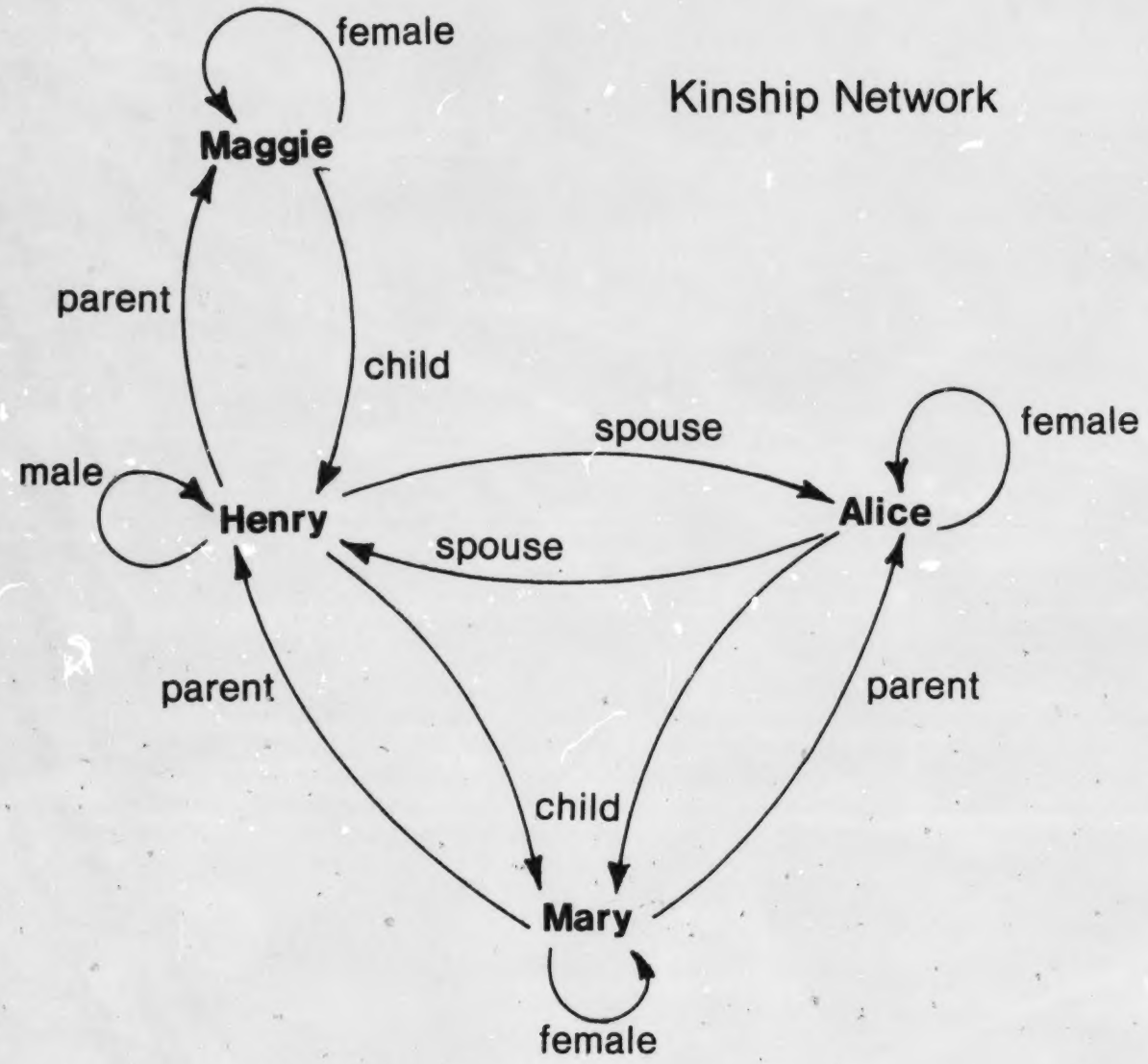

Figure 5. An example of a piece of a network encoding knowledge about kinship relations. The network consists of a set of nodes representing people and a set of arcs representing the basic relationships among people. Only three different arc types are required to represent the kin relations and two to represent the sex of the individuals. These are CHILD, PARENT, SPOUSE and MALE and FEMALE, respectively. Special procedures can then be defined to operate on such a network to determine the kinship relation which holds among any two individuals in such a network.

Now consider, as an example, the following procedural definition of a function which produces as its result the set of all parents of individual :x.

```
define parent(:x)
     return nodeset with "child" to :x.
```

This function merely returns, as a result, the set of nodes which have a pointer labeled "child" to node :x. The network representation of this procedure is given in Figure 6. One could then define "child" by analogy with "parent",

```
child(x) is-like "parent" with "parent" for "child".
```

The appropriate definition of "child" is then constructed by creating a new function which is a copy of the old, except that for every occurrence of "child" in the original, the term "parent" is put in its place. This would produce a function which would return the set of nodes accessible through the pointer "parent." In the framework illustrated in Figure 5, this would be a correct procedure for producing the set of children for some individual :x. Now the procedure NODESET is defined so that if the variable :x is filled by a set of nodes, rather than by a node for a single individual, it will generate a set which contains all of the nodes that can reach any of the nodes in question through the named pointer (e.g. "parent" or "child"). Thus, the function FEMALE defined by analogy with PARENT as:

```
female is-like "parent" with "female" for "child".
```

will return a set containing those elements of its argument set which represent a female. Thus, we can define MOTHER as

```
define mother(:x)
     return female parent :x.
```

Then, assuming the functions MALE and SPOUSE (which could, of course, be defined by analogy with FEMALE), we could create the functions, FATHER, SON, DAUGHTER, GRANDPARENT, etc. by using the following analogies. These procedures can be created by noting the following relationships:

```
father is-like "mother" with "male" for "female".

son is-like "father" with "child" for "parent".

daughter is-like "son" with "female" for "male".

grandparent is-like "parent" with parent(:x) for :x.
```

With a little care, procedures to produce the entire set of English kinship terms can be readily constructed, by analogy, from two basic procedures.

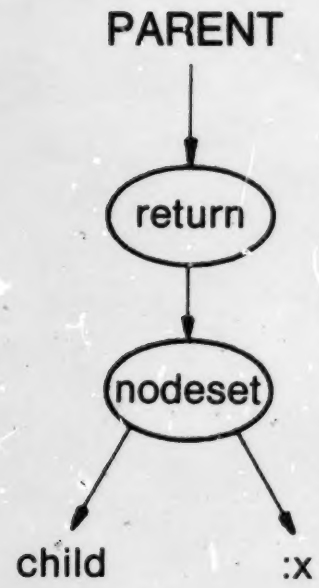

Figure 6. Network representation for PARENT. The function NODESET takes two arguments, an arc name (in this case "child") and a set of individuals (in this case the variable :x). It then returns as a result the set of nodes in the data base which have the specified arc pointing to any of the set :x.

One interesting observation to be made about the procedures thus created is that there are a number of possible analogies which will create procedures which carry out the same task, but, depending on the particular analogies used, different ways of computing the same things will be employed. For example, we could say that:

grandmother is-like "mother" with "grandparent" for "parent".

or we could say:

grandmother is-like "mother" with parent(:x) for :x.

These two ways of defining GRANDMOTHER correspond to two conceptions of a grandmother, one in which grandmother is conceived of as the female of the grandparents as the mother is the female of the parents, and another in which she can be conceived as one who differs from a mother by being the parent not of the individual in question, but of the parent of that individual. The network representations of these two different GRANDMOTHER procedures as shown in Figure 7. It may well be that not only are analogies important in the initial teaching of a concept, but they may also be useful for teaching alternate conceptualizations. It may well be that this is a primary role of metaphor.

In all of our examples so far, we have assumed that the relevant dimensions of modification were already known to the system. In general, of course, we do not know the relevant dimensions of comparison. It is to point out the relevant dimensions that four term analogical relations are important. Consider the following four term analogy:

grandfather is-to "grandmother" as "father" to "mother".

This statement will cause a new GRANDFATHER procedure to be created in the following way: first the structures for FATHER and MOTHER are compared and their differences are found. In this case, they differ only in that where MOTHER uses the procedure FEMALE, FATHER uses the procedure MALE. This set of differences can then be applied, through the IS-LIKE mechanism, to GRANDMOTHER, finally creating FATHER. Note that this procedure will work whichever of the conceptualizations of GRANDMOTHER had been chosen.

In general, this process of matching pairs of procedures to find their differences is very similar to the matching processes in MERLIN and, like MERLIN, is generally not deterministic. Depending on exactly how the differences between pairs of procedures are characterized, many different mapping functions can be found. Each of these mapping functions represent a way of characterizing the difference between a pair of procedures. If, like this example, the original procedures are rather close together, the process of extracting differences will be relatively straight forward. In other cases, for example the difference between MOTHER and SQUARE, the differences will be relatively complex, and an analogy probably cannot usefully be drawn between them.

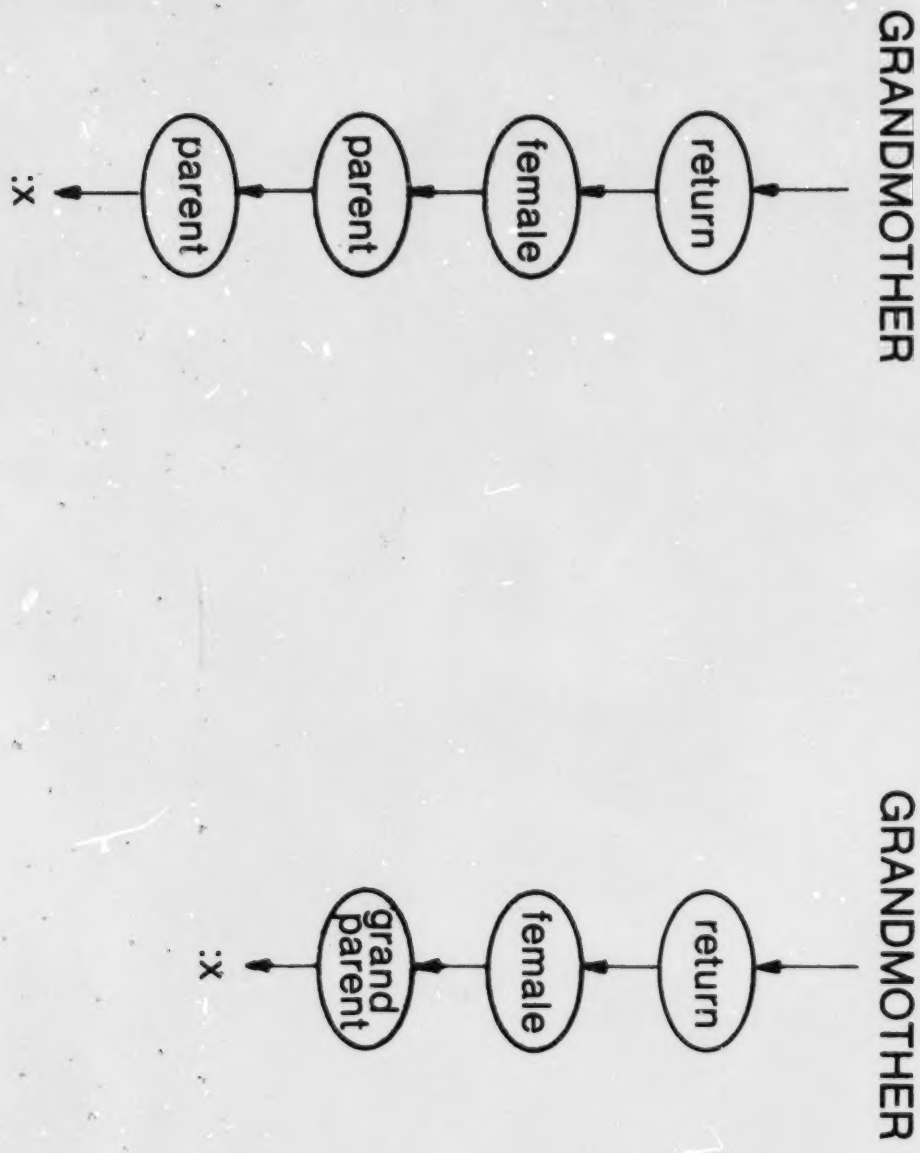

Figure 7. Network representations for two different versions of the GRANDMOTHER procedure. Both procedures produce the same results, they simply do it in different ways.

Of course, the examples discussed above are not intended to represent the particular knowledge about squares, parents, or grandparents that people actually have. Rather, they are intended as mere demonstrations of the sorts of processes which can be employed to create new schemata from old ones. Once created, the new schemata no longer depend on the schemata from which they were spawned, but are full-fledged procedures in their own right with all of the features of procedurally represented knowledge. Nevertheless, a number of schemata, all spawned in different ways from the same schema, will share a good deal of common structure, and it is possible to compare pairs of them to find the pattern of modifications required to get from one to the other.

Analogical Extensions of Lexical Meanings

We believe that the sort of processes outlined above play an important role in our learning of new concepts. It seems especially interesting to consider some of the analogies which can be drawn among the meanings of various classes of verbs. It appears that often, as with the analogy involving "son" and "daughter", relatively simple differences occur among verbs which are consistent with the idea that verb meanings may have been generated by analogy from a few basic underlying verb types.

In the language comprehension system we built in our Active Semantic Network formalism, Rumelhart and Levin (1975) showed how a simple procedural definition could be given to various verbs such that these verbs, when encountered in a text, would determine whether the facts (or some part of the facts) being communicated by the verbs were already known. If not, they would create a memory representation of the relevant facts and inferences. One of the verbs we defined was the verb "move" (intransitive sense). We suggested that move could be defined roughly as follows:

```
define move(:x,from :y to :z)
     means change(from loc(:x,:y), to loc(:x,:z)).
```

Similarly, we defined the verb "get" to be roughly:

```
define get(:x,:y,from :z)
     means change(from possed-by(:y,:z) to possed-by(:y,:x)).
```

It can be seen that "get" can easily be from "move" by the analogy:

get is-like "move" with "possed-by" for "loc" :x for :z, :y for :x and :z for :y.

Jackendoff (1975) produced a rather interesting set of examples illustrating large sets of verbs whose meanings are related in just the same relatively simple sorts of ways as the familial relations. Thus, for example, Jackendoff argued that the verb "keep" in the positional sense (e.g. Bill kept the book on the desk.) and in the possessional sense (e.g Bill kept the book.) differ in much the same ways we suggested for "move" and "get". Jackendoff showed that a rather large array of verbs and verb meanings could be related to one another by relatively simple analogical relationships.

Analogical Processes in Learning a Text Editor

For several years now, we have, with several of our colleagues carried out a series of studies aimed at understanding what we have called "complex learning" (cf. Bott, 1978; Norman, Gentner & Stevens, 1976; Norman (1975); Norman, 1978; Norman & Gentner 1978, Norman, 1980). We

sought to study topics which required several hours, rather than several minutes or several weeks to learn. We studied a variety of different topics. Ultimately, we focused most of our attention on observing people while learning to use a text editor.

The particular text editor available on our laboratory is the Ed text editor available under the UNIX operating system. In our experimental situation, we asked students to learn how to use the text editor by actually using it, referring to an instructional manual for guidance. In the examples that follow, we were using a very simple manual that we wrote. The basic experimental situation is shown in Figure 8. The student sat in the booth, typing material to Ed on a computer terminal. The instruction manual was displayed to the student a paragraph at a time on a second terminal. All keystrokes, along with their interstroke intervals, were recorded by the computer. In addition, an observer sat in the room with the student and occasionally asked questions or asked the student to think aloud during portions of the learning period. Each session was tape recorded.

An experimental situation such as this generates an enormous quantity of data. We have analyzed numerous segments of the learning protocol. In this paper, we will focus on a typical example which illustrates how the sorts of analogical processes discussed in the previous section show up in such learning situations. At the start, the Ed screen was always blank except for a cursor. The student began by reading a basic introduction to text editing on the instruction terminal. Then, an attempt was made to teach the specific commands used by Ed. Students were given the following instruction on the instruction terminal:

> You are going to learn how to print the text on the screen.
>
> Type
>
> 3p
>
> Type the key marked RETURN

Most students typed this sequence without difficulty, and the message illustrated in Figure 9 appeared on the screen. The first line on the screen is the command typed by the student (3p). The second line is the resulting output from Ed, and the third line is the cursor.

We might imagine that as a result of this experience the student would create an internal representation of the event similar to that shown in Figure 10. Here we have a little procedure for printing text line three-- pressing the keys 3 and p causes line 3 to be printed on the screen.

The next part of the instruction manual was built on the following statement:

> Now try printing the fifth line.

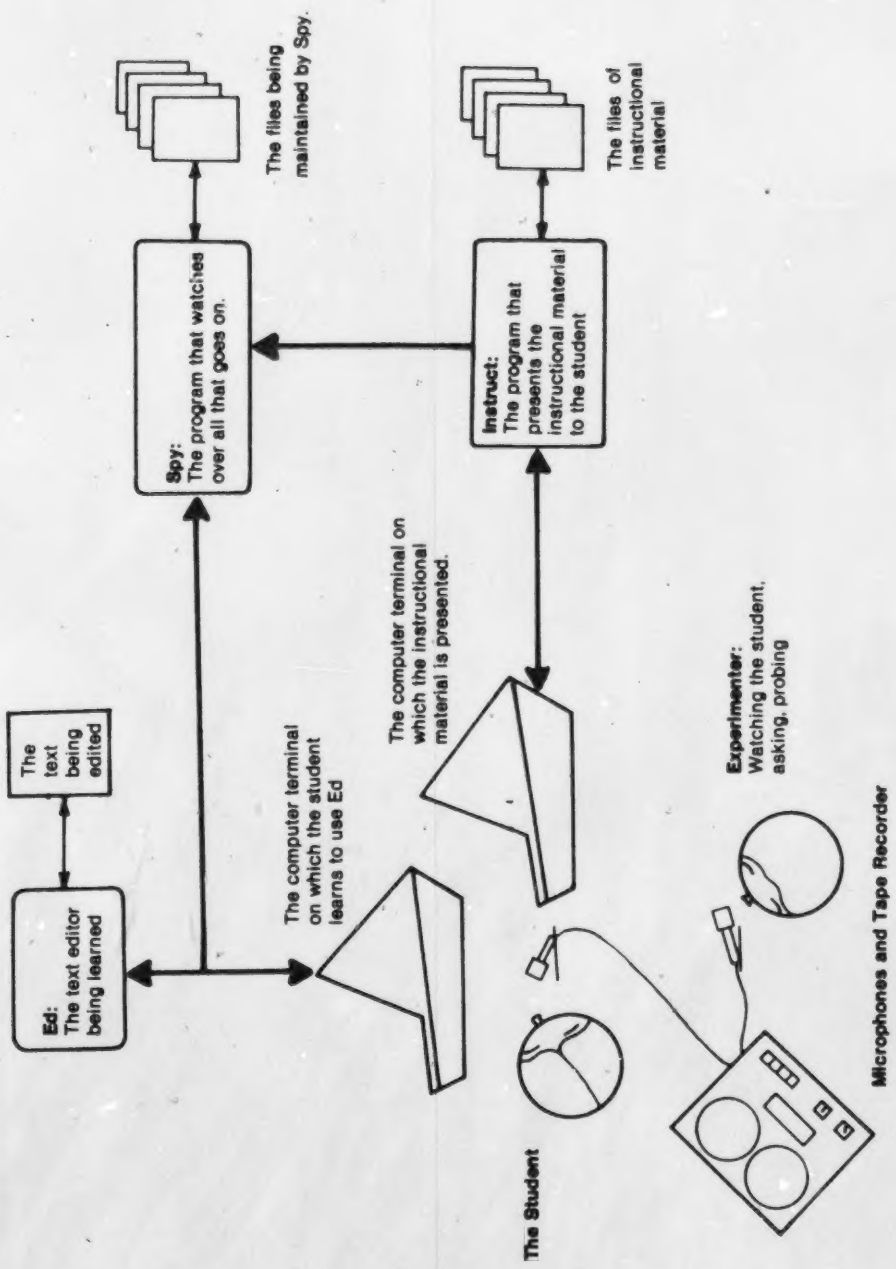
The files being maintained by Spy.
The files of instructional material
Spy: The program that watches over all that goes on.
Instruct: The program that presents the instructional material to the student
The computer terminal on which the instructional material is presented.
The computer terminal on which the student learns to use Ed
The text being edited
Ed: The text editor being learned
Experimenter: Watching the student, asking, probing
Microphones and Tape Recorder
The Student

Figure 8. Basic experimental situation for observing students learning Ed. The student sat in a booth before two computer terminals. One terminal was used to give commands to Ed and carry out the text editing task. The other terminal was used to instruct the students on the editor and was controlled by a INSTRUCT, an interactive program for teaching. All interaction with either Ed or INSTRUCT was monitored and recorded by another program SPY. An experimenter sat in the booth with the student and occasionally asked questions. All conversation was tape recorded.

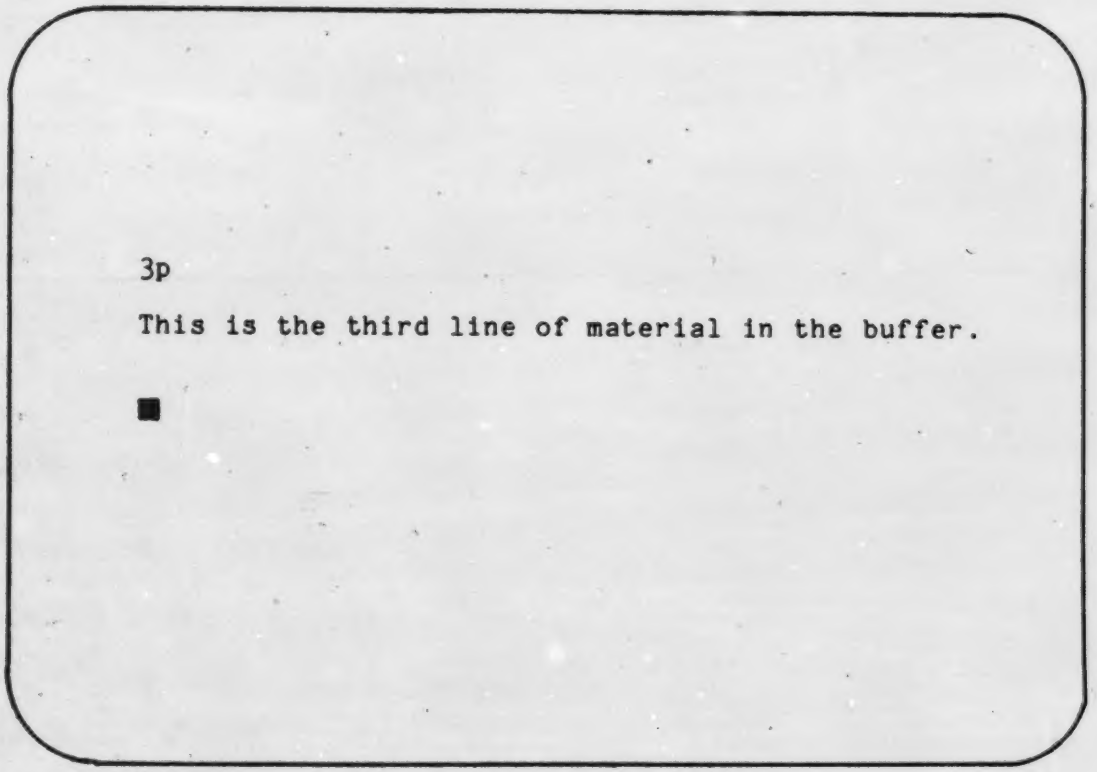

Figure 9. The contents of the terminal screen following a command to type the third line of the text.

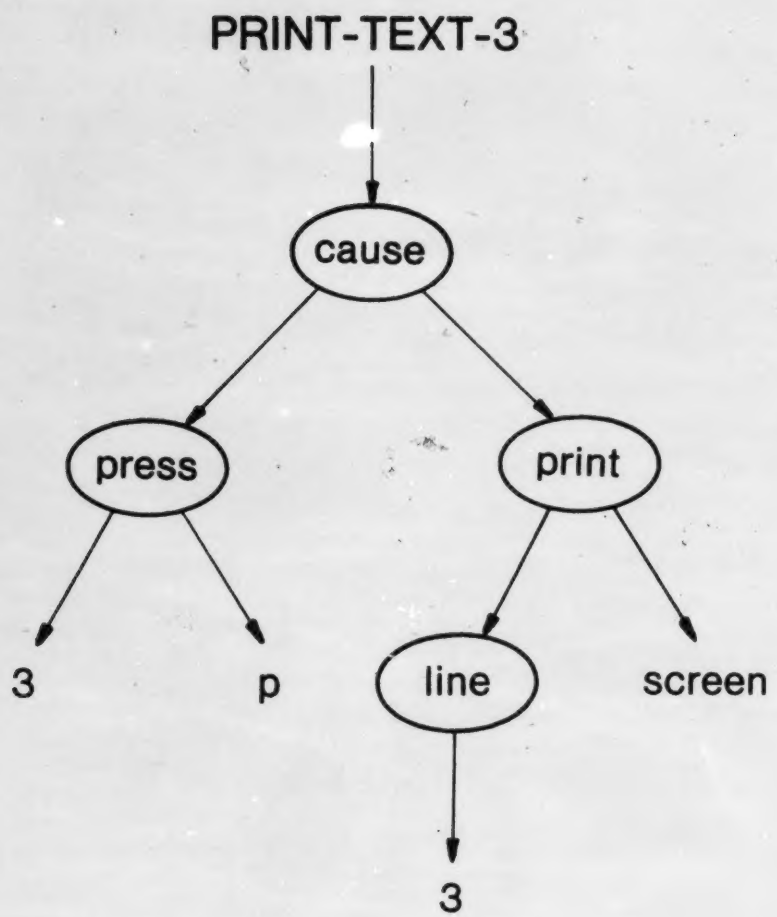

Figure 10. Representation of a procedure PRINT-TEXT-3 which we suppose may have been created as a result of the instruction to print out the third line of the text.

Clearly, this statement requires that the student learn by analogy. We can imagine that this command would be interpreted by the student as:

print-text-5 is-like "print-text-3" with 3 for 5.

This procedure would, of course, work and produce a procedure exactly like that of figure 10 except for the 5 replacing the 3 in the figure. Presumably, the student could also have made the inference from this experience that

print-text is-like "print-text-3" with :n for 3.

This would produce the general program for printing any line of text illustrated in Figure 11.

Somewhat later in the session, students were taught to understand the "delete" command. The text of the beginning of the lesson on "delete" from the instruction manual for Ed is given below:

> Suppose we want to get rid of extra lines in the buffer. This is done by the delete command "d". Except that "d" deletes lines instead of printing them, its action is similar to that of "p".

This text is an invitation to build a structure for "delete" by analogy with that for print. According to the model we have been discussing, we might imagine that the student would interpret this as follows:

> delete-text is-like "print-text" with "d" for "p and "delete" for "print".

This would lead to the structure illustrated in Figure 12. There is some evidence that our students actually constructed a schema similar to this for delete.

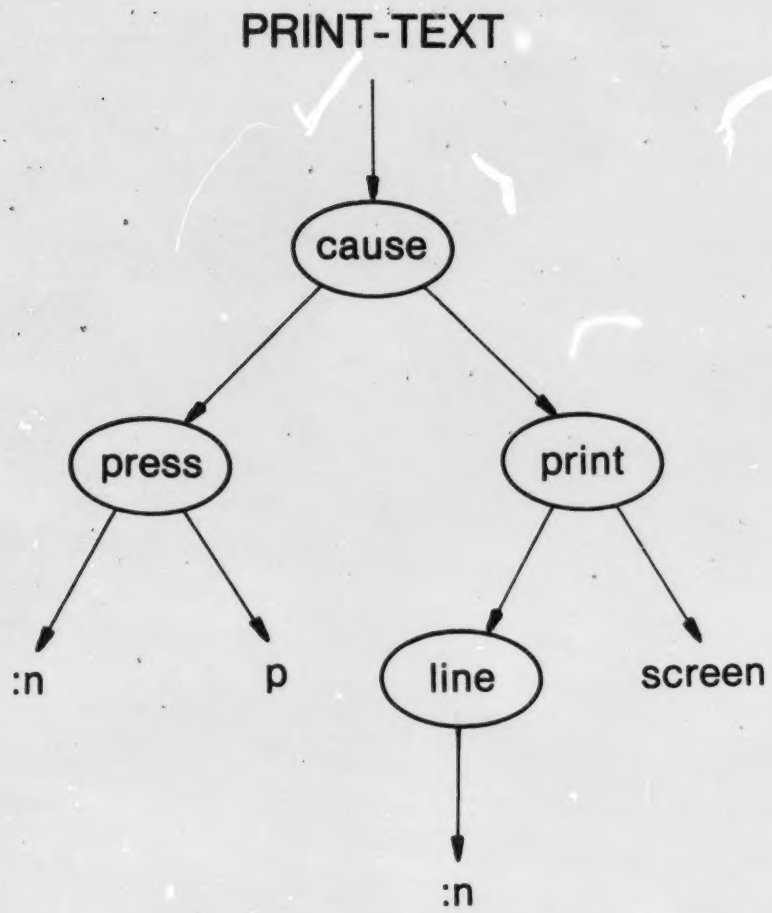

Figure 11. Network representation of a procedure for printing out any line of a text.

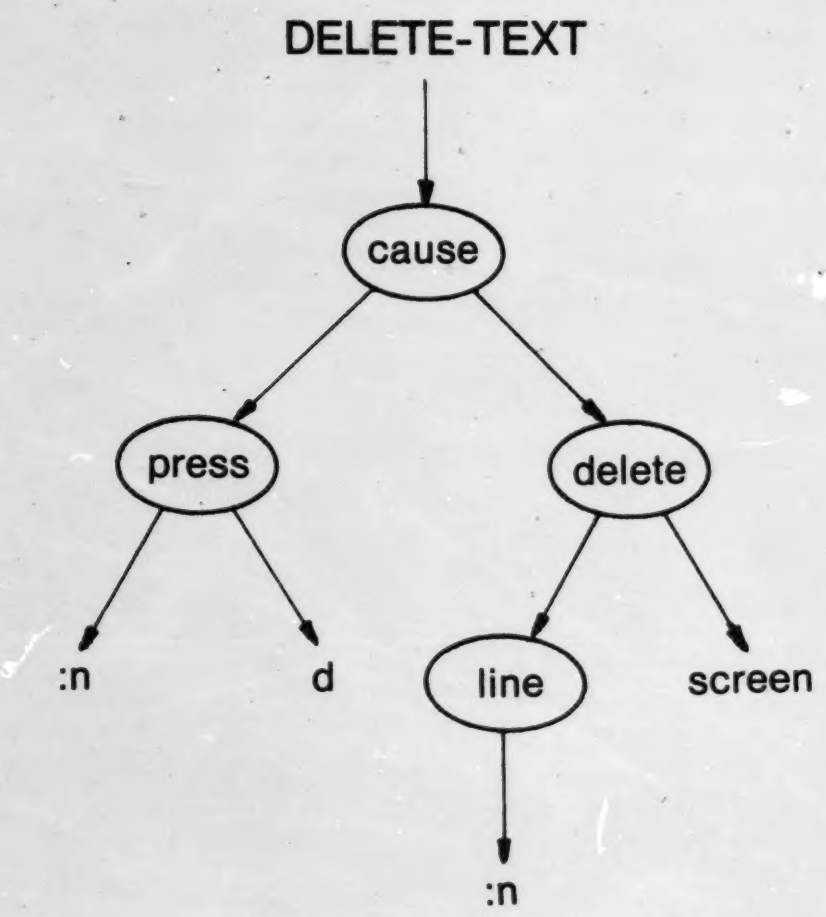

Figure 12. Network representation of procedure DELETE-TEXT which is derived by analogy from the general PRINT-TEXT procedure.

In one example, after receiving instruction on the deleting lines from a buffer, the student was asked to delete line 4. At this time, the screen contained a number of lines of text, including line 4. According to the delete schema illustrated in the figure, the student should type 4d. This was done. However, the schema also predicted that line 4 should be deleted from this screen. It was not. After typing "4d" and seeing nothing happen, the student sat staring at the screen of the terminal, and then looking back and forth from the instruction manual to the screen. The experimenter, sitting in the experimental booth with the student, asked the student to explain the problem:

Experimenter: What did you just do?

Student: I deleted line 4, at least I was thinking I was deleting line 4.

Experimenter: What did you expect to happen?

Student: I expected line 4 to disappear, either that or the text to be reprinted without line 4 in it.

Experimenter: Uh-huh, but that didn't happen.

Student: It didn't happen.

A common response of students was to assume that somehow or other Ed didn't "notice" the command, so they typed "4d" once more. This action invoked the delete command a second time, thereby eliminating in the buffer the new line 4, which used to be line 5.

Although this analysis fits rather neatly into the model we have been describing, the situation is really more complex and points to additional constraints on how students will create analogies. The error committed by the students was in part a result of their incomplete conceptualization of the various parts of the computer system. They reasoned that the screen was a sort of window on the computers knowledge, so if a line was deleted from the computers memory, it should no longer be visible on the screen. These same inferences did not occur when the very same instruction manual and editor were used on a hard copy terminal. Here the student's model of the relationship between the paper and the computer's knowledge were very different. They found it easy to see the paper as a medium on which the computer typed commanded messages. They knew the computer could not physically erase a line previously printed and thus interpreted the description of the delete command differently. The difference in the kinds of mental models that students bring to the situation clearly play a critical role in the kinds of analogies students will employ. It is a far more important role than that of the formal instruction received.

This was only one of the many problems that our students had in attempting to understand the operation of the text editor. We found

that although students made many errors in learning to use the editor, their errors were not random. Rather, they almost always were responding in terms of a plausible interpretation of what they were told. They created models and made plausible inferences by analogy with situations they already understood. We found that before we could really teach them to understand the operation of the text editor in general, and the delete process in particular, a rather different approach was required.

To make Ed understandable, we needed to give the students analogical frameworks more appropriate than the ones they naturally used. The difficulty, however, was that our students knew nothing of computers, so either our model was going to be incomplete or we were going to have to spend considerable time giving them a complete model. We discovered an interesting solution to this dilemma: give many different conceptual models, each one simple, each making a different point.

We developed three distinct models which, together, seemed to offer a reasonable account for the various aspects of a text editor. We developed the "secretary" model, the "card file" model and the "tape recorder" model. The secretarial model explains some aspects of Ed, especially the overall format of intermixing commands and textual material. The difficulty with this model, however, was that our students expected Ed to be as intelligent and understanding as a real secretary would be. Hence, if they gave the append command, they then fell prey to what we have called the append-mode trap. When they finished appending test, they would issue a command and expect Ed to carry it out. Instead, Ed would treat the command as another line of text and simply add it to the file. But, because Ed often gets commands and follows them without giving any visible reaction, the students were sometimes unaware of what happened. Presumably, a real secretary able to distinguish between the text being taken in dictation and the interspersed comments about the format of the letter etc. Ed takes everything literally and has to be told explicitly to suspend dictation and register a command etc.

Therefore, the secretarial model has some virtues and some difficulties. The tape recorder model helps students understand the append-mode trap. Think of Ed as a tape recorder and the append command as equivalent to recording on the tape recorder. Once a tape recorder has been put into record mode, it faithfully records every sound that reaches its microphones. The only way to stop the recording is to perform the explicit action that terminates the record mode (usually by pushing the lever marked "stop").

The tape recorder model has the virtue of explaining about the append-mode trap, but it is deficient in explaining the delete command. The filing card model offers a good analogy for understanding the line-oriented structure of the recorders kept by Ed. Thus, the renumbering of lines that takes place after a delete or append command is completed is easy to interpret, given the model of the removal or addition of cards in the file. Clearly, the filing card model by itself does not explain why the deleted line is not removed from the text the student

sees on the screen, but it does provide the proper conceptual framework. An appropriate interpretation of the situation is that the contents of the file cards are not visible to the user of Ed. Those are Ed's private files. If you want to know what is in the files, you must ask to see them with a "print" command.

The need for three separate models is reminiscent of the case of teaching fractions. None of the "pure" models are perfect. Each has its own advantages and disadvantages. Apparently, what happens as we become expert in a domain is that we become better and better at choosing the appropriate model for the situation at hand. The success of such models in teaching are, we believe, an essential clue to the normal learning process. Students appear to create their own models if not given any such guidance. A major pedogogical issue here is that a student's own creations are often suprisingly good at providing an explanation of what has been happening. Thus, neither student nor instructor realizes how bad the model is, and it is not until the model leads to some major difficulty that the hint of trouble develops.

Conclusions

We have adopted the view that much of our knowledge exists embedded in specialized procedures which are employed in the interpretation events in our environment. We call these packets schemata. One problem with such a view is that it is difficult to see how such procedures can be built up through experience. How can we create new schemata? We have proposed that complex new procedures can be readily created by modeling them on existing schemata and modifying them slightly. We believe that the typical course of such a learning process consists of an initial creation of a new schema by modeling it on an existing schema. This new schema, however, is not perfect. It may occasionally mispredict events and otherwise be inadequate. We then believe that the newly acquired schema undergoes a process of refinement which we have dubbed tuning. We have not addressed the tuning problem in this paper. Instead we have focused on this process of modeling one schema on another. We believe that this modeling process is properly called learning by analogy.

We find examples of learning and teaching by analogy to be absolutely ubiquitous. It appears that the usual learning sequence proceeds as follows: Whenever one encounters a new situation they seek to interpret it in terms of existing schemata. If they succeed, they understand the situation and no new schemata need be created. Occasionally, however, there are no existing schemata which can offer a satisfactory account of a situation. In this case, we assume that the next best schemata are found. Presumably, since no completely applicable schemata existed, the schemata used to interpret the input had regions of mismatch with the input situation. In some cases, essential features of the interpreting schemata might not be present with other features in their place. Presumably, such a situation serves as a trigger for the creation of a new schema. The schema applied inappropriately to the current situation can thus serve as the source domain and thus as a

model from which to generate the new schema. The ways in which the inappropriate schema is inappropriate give an initial set of differences by which the new schema is different from the old. Importantly, those characteristics of the new schema which are not contradicted by the new situation are assumed to be carried over into the new domain, even though they are not specifically apparent in the initial learning situation. It is through such carrying over that the analogical process is both powerful and prone to error. Carrying over existing features of existing schemata allow us to make inferences about the new situation without explicit knowledge of the new situation. It allows us to learn a good deal very quickly. It also can lead to error. If the analogy is a good one, most of the inferences we make will be appropriate. On the other hand, some of them will be incorrect. It is these incorrect inferences which can allow us, as analysts, to see the features of the source schemata in a subject's performance on a new domain.

There are, we believe, a number of instructional implications of the view of learning we have been developing. In particular, it suggests that the appropriate way to teach a domain is to provide the student with a conceptual model which has the following properties:

(1) It should be based on a domain with which the student is very knowledgeable and in which the student can reason readily.

(2) The target domain and the source domain should differ by a minimum number of specifiable dimensions.

(3) Operations which are natural within the target domain should also be natural within the source domain.

(4) Operations inappropriate within the target domain should also be inappropriate within the source domain.

Typically, no single model will suffice for any reasonably complex subject matter. In such cases, a set of models, each with their specifiable domains of applicability, are often useful. Ultimately, several schemata may be created for any given domain, each with their own, built-in, context dependencies determining when each one is applicable. Each of these schemata can be considered alternate conceptualizations of the target domain.

References

Bobrow, D. G., & Norman, D. A. Some principles of memory schemata. In D. G. Bobrow & A. M. Collins (Eds.), Representation and understanding: Studies in Cognitive Science. New York: Academic Press, 1975.

Bott, R. A. A study of complex learning, theory and methodologies. Unpublished doctoral dissertation, University of California, San Diego, 1978.

Jackendoff, R. A system of semantic primitives. In R. Schank & B. L. Nash-Webber (Eds.), Papers from the conference on theoretical issues in natural language processing (TINLAP-1), Cambridge, Mass.: June 1975.

Moore, J., & Newell, A. How can MERLIN understand? In L. W. Gregg (Ed.), Knowledge and cognition. Potomac, Md.: Erlbaum Associates, 1973.

Norman, D. A. Learning and teaching. In P. M. A. Rabbitt & S. Dornic (Eds.), Attention and performance V. Proceedings of the Fifth Symposium on Attention and Performance, Stockholm, Sweden. London: Academic Press, 1975.

Norman, D. A. Notes toward a theory of complex learning. In A. M. Lesgold, J. W. Pellegrino, S. Fokkema, & R. Glaser (Eds.), Cognitive psychology and instruction. New York: Plenum Publishing Co., 1978.

Norman, D. A. Teaching, learning, and the representation of knowledge. In R. E. Snow, P. A. Frederico, & W. E. Montague (Eds.), Aptitude, learning, and instruction. Volume 2: Cognitive process analyses of learning and problem solving. Hillsdale, N.J.: Lawrence Erlbaum Associates, 1980.

Norman, D. A., & Gentner, D. R. Human learning and performance. Naval Research Reviews, 1978, 31, (9), 9 - 19.

Norman, D. A., Gentner, D. R., & Stevens, A. L. Comments on learning: Schemata and memory representation. In D. Klahr (Ed.), Cognition and instruction. Hillsdale, N. J.: Erlbaum Associates, 1976.

Norman, D. A., Rumelhart, D. E., & the LNR Research Group. Explorations in cognition. San Francisco: Freeman, 1975

Rumelhart, D. E., & Levin, J. A. A language comprehension system. In D. A. Norman, D. E. Rumelhart, & The LNR Research Group, Explorations in cognition. San Francisco: Freeman, 1975.

Rumelhart, D. E., & Norman, D. A. Active semantic networks as a model of human memory. Proceedings of the Third International Joint Conference on Artificial Intellegence. Stanford, California, 1973.

Rumelhart D. E., & Norman, D. A. Accretion, tuning and restructuring: Three modes of learning. In J. W. Cotton & R. Klatzky (Eds.) Semantic factors in cognition. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1978.

Rumelhart, D. E., & Ortony, A. The representation of knowledge in memory. In R. C. Anderson, R. J. Spiro, & W. E. Montague (Eds.), Schooling and the acquisition of knowledge. Hillsdale, N. J.: Erl baum Associates, 1977.

Rumelhart, D. E. Schemata: The Building Blocks of Cognition In R. Spiro, B. Bruce and W. Brewer (eds.), Theoretical Issues in Reading Comprehension. Hillsdale, N.J. : Lawrence Erlbaum Assoc., in press.

Scragg, G. W. Answering questions about processes. In D. A. Norman, D. E. Rumelhart, & The LNR Research Group, Explorations in cognition. San Francisco: Freeman, 1975.

Wason, P., & Johnson-Laird, P.N. Psychology of reasoning: Structure and content. Cambridge: Harvard University Press, 1972.

Winograd, T. Frame representations and the declarative-procedural controversy. In D. G. Bobrow & A. M. Collins (Eds.), Representation and understanding: Studies in cognitive science. New York: Academic Press, 1975.